于新海

和益达-龙湖项目开发组

2012.3



架构开发文档

目录

[MyWeb开发规范 3](#_Toc331144956)

[术语定义： 3](#_Toc331144957)

[Mvc开发规范 3](#_Toc331144958)

[枚举规范 6](#_Toc331144959)

[C#（服务器）端开发API 9](#_Toc331144960)

[代码生成器 9](#_Toc331144961)

[Web（浏览器）端定义 9](#_Toc331144962)

[开发标准 9](#_Toc331144963)

[系统定义的Css Class 10](#_Toc331144964)

[系统函数 12](#_Toc331144965)

[系统处理的扩展方法 13](#_Toc331144966)

[扩展的 jQuery函数 14](#_Toc331144967)

[数据表格（FlexiGrid）： 15](#_Toc331144968)

[圆角方案： 17](#_Toc331144969)

[TextHelper方案： 17](#_Toc331144970)

[Boxy方案： 18](#_Toc331144971)

[客户端弹出窗口方案 19](#_Toc331144972)

[弹出查询 21](#_Toc331144973)

[验证 22](#_Toc331144974)

[汉字转化为拼音 23](#_Toc331144975)

[MyOql 24](#_Toc331144976)

[1. 生成实体配置 25](#_Toc331144977)

[2. 数据库名称与程序名称映射 25](#_Toc331144978)

[解决方案 25](#_Toc331144979)

[权限方案: Udi 25](#_Toc331144980)

[排序方案 26](#_Toc331144981)

[表锁 29](#_Toc331144982)

[批量数据推送 30](#_Toc331144983)

[接口开发 31](#_Toc331144984)

[事务 31](#_Toc331144985)

[报表开发 33](#_Toc331144986)

[系统服务 34](#_Toc331144987)

[数据导出 34](#_Toc331144988)

[异步加载树列表 35](#_Toc331144989)

[跨页面传值（按查询的导出方案） 36](#_Toc331144990)

[模拟Session方案 37](#_Toc331144991)

[数据库系统定义 37](#_Toc331144992)

[表定义 37](#_Toc331144993)

[视图定义 37](#_Toc331144994)

[函数定义（不包含程序集定义的函数） 37](#_Toc331144995)

[存储过程定义 38](#_Toc331144996)

[程序集定义 39](#_Toc331144997)

[影子库方案 40](#_Toc331144998)

[动态SQL方案 40](#_Toc331144999)

[代码规范 42](#_Toc331145000)

[文件结构 42](#_Toc331145001)

[命名规范 43](#_Toc331145002)

[代码格式 43](#_Toc331145003)

[公共页面 43](#_Toc331145004)

## MyWeb开发规范

### 术语定义：

|  |  |
| --- | --- |
| 列表 | 表示页面显示的针对某一主实体的列表 |
| 卡片 | 表示针对某一主实体的详细说明 |
| 引用 | 在页面上引用一个实体时，先弹出一个关于该实体的列表，选择某实体后，返回对该实体的引用，实体引用需要指定唯一ID和显示Name |
| 实体 | 指ORM中映射的对象 |
| 模型 | 指 Mvc 里绑定的对象 |
|  |  |
|  |  |

### 开发规范(Mvc)

1. 在定义URL 中, 尽量使用 Area/Controller/Action.aspx 三级形式, 如果非常有必要, 采用uid的四级,否则, 其它参数都通过QueryString进行传递. 这样做的好处是Mvc定义Action时, 明确表示参数函义:如: Host/User/List.aspx?RoleId=Admin 和 Host/User/List.aspx?DeptID = 1 ；而且，Actiion 参数名与 QueryKey相对应。
2. 关于母版页，由于页面可以被其它页面当成控件加载，而外面的容器结构会和控件结构不同，所以需要在母版页中清晰界定出内容区如 class=”FillHeight”，使容器在加载时指定。

由于页面生成器规范，需要在母版页引用 MyJs\_{Area}.js ，并且需要设置内容区 class=”Main”

1. Model，不建议使用继承，建议使用组合类（不绝对）。建议使用 MyOql生成的实体进行组合， Model的构造函数要对该组合的子对象进行初始化。

如果Model类中有非元类型对象，则需要在默认构造函数中初始化，否则可能在直接引用该属性时出错。

客户端Post的键 与服务器端Model对应关系：

1. 前缀 = Model 的变量名
2. 数组用 [index] 表示，且 index 必须连续。
3. 子对象用 “.” 连接。

如：（book 前缀可指定，也可以不指定）

客户端Post的数据： book[0].Id = 1 , book[0].Name=”鲁迅” , book[1].Id=2 , book[1].Name=”朱德”

服务器端定义: public ActionResult Recv(List<BookModel> book)

1. Mvc的参数 uid 或 id 并不能表示实际的业务含义，它可以是 UserId ，也可能是 FeeId。我们不能改变Action 的参数名。所以需要在Action 里再次指定。

//之后使用 UserId ，避免使用 uid 这样的含义不清的变量。

var UserId = uid ;

1. 系统Card页面(key项)颜色提示规范
2. 系统带入项： 灰色，使用类class=”withgray”
3. 必填项： 红色，使用类class=”withred”
4. 选填项： 黑色，系统默认色
5. 开发不用处理 try catch , Controller 层处理Session，参数，返回值，错误。
6. 使用RenderJsVar 服务器向客户端输出Js。
7. 尽量使用循环替代递归。尽量控制递归的级数。

### 枚举规范

如果数据库中的字段不可为空，那么查询条件应该必查。

如果数据库中的字段可空， 那么查询条件应该可空。则该枚举从1开始。 0表示Null，预留给查询，以表示用户没有选择。

在非标准情况下，系统使用了0 表示某一个函义，查询条件项要判断，如：YesNoEnum 系统使用 No=0，Yes=1，系统Post到服务器端，~~Model绑定会使 没有选择的项和No项混淆。Model 采用 IsNullable<T> 类型~~。枚举项要表示该属性值为空，按以下规则：

1. 当枚举没有定义 0 的时候， 空值表示 0
2. 当枚举定义了 None 时，空值表示 None
3. 当枚举定义了 Default 时，空值表示 Default
4. 当枚举没有定义了 -1 时， 空值表示 -1
5. 当枚举没有定义 int.MinValue 时，空值表示 int.MinValue
6. 如果枚举继自 ushort ，或 uint 之类的。 该枚举一定不要定义0 值。
7. 其它情况，系统抛出错误。

系统枚举：

1. YesNoEnum { Yes = 1 , No = 2 }
2. IsAbleEnum { Enable=1 , Disable=2 }

**扩展方法**

为了方便调用，把常用枚举扩展常用方法，如

MySessionKey.UserID.Get();

ConfigKey.InitPassword.Get () ;

#### 枚举类

在某些情况下，如需要表现更多信息，可以采用类表示枚举。定义枚举类的方式：

* + - 1. 先定义枚举项类型，继承自 EnmItem 泛型类。
      2. 定义枚举项类型的 TypeConverter
      3. 定义枚举类

public class AmountRoundedEnum : IReadEntity

{

[TypeConverter(typeof(AmountRoundedEnumConverter))]

public class \_ : EnmItem<AmountRoundedEnum, float>

{

public \_(float code, string name)

: base(code, name)

{

}

}

public static \_ Zero { get { return new \_(0.00f, "数量不作取整"); } }

public static \_ Half { get { return new \_(0.50f, "数量取整到0.5"); } }

public static \_ One { get { return new \_(1.00f, "数量取整到1"); } }

public static \_ Five { get { return new \_(5.00f, "数量取整到5"); } }

public static \_ Ten { get { return new \_(10.00f, "数量取整到10"); } }

public object GetPropertyValue(string PropertyName)

{

if (PropertyName == "Zero") return Zero;

if (PropertyName == "Half") return Half;

if (PropertyName == "One") return One;

if (PropertyName == "Five") return Five;

if (PropertyName == "Ten") return Ten;

return null;

}

public string[] GetProperties()

{

return new string[] { "Five", "Half", "One", "Ten", "Zero" };

}

}

public class AmountRoundedEnumConverter : TypeConverter

{

public override bool CanConvertFrom(ITypeDescriptorContext context, Type sourceType)

{

if (sourceType.FullName == "System.String") return true;

if (sourceType.FullName == "System.DBNull") return true;

if (sourceType.FullName == "System.Decimal") return true;

if (sourceType.FullName == "System.Single") return true;

return base.CanConvertFrom(context, sourceType);

}

public override bool CanConvertTo(ITypeDescriptorContext context, Type destinationType)

{

if (destinationType.FullName == "System.String") return true;

if (destinationType.FullName == "System.DBNull") return true;

if (destinationType.FullName == "System.Decimal") return true;

if (destinationType.FullName == "System.Single") return true;

return base.CanConvertTo(context, destinationType);

}

public override object ConvertFrom(ITypeDescriptorContext context, System.Globalization.CultureInfo culture, object value)

{

if (value.IsDBNull()) return null;

var str = value.AsFloat();

AmountRoundedEnum.\_ retVal = null;

var ent = new AmountRoundedEnum();

ent.GetProperties().FirstOrDefault(o =>

{

var obj = ent.GetPropertyValue(o) as AmountRoundedEnum.\_;

if (obj == null) return false;

if (obj.Code == str)

{

retVal = obj;

return true;

}

return false;

});

return retVal;

}

public override object ConvertTo(ITypeDescriptorContext context, System.Globalization.CultureInfo culture, object value, Type destinationType)

{

if (value.IsDBNull()) return null;

if (destinationType.FullName == "System.String") return value.AsString();

if (destinationType.FullName == "System.Decimal") return value.ToString().AsDecimal();

if (destinationType.FullName == "System.Single") return value.ToString().AsFloat();

return base.ConvertTo(context, culture, value, destinationType);

}

}

其中 EnmItem的Code 可表示某一项值， 中文仅显示。 每一项属性用于在程序中做为变量的判断值如：

if(userPayType == PayType.Online ) { … }

绑定到TextHelper 的方法，使用：

IEnumRadio , TextHelperIEnum (参见MyCon 源码。)

### 自定义配置项的读取

在 Web.Config 文件定义配置项。

配置项枚举化，枚举定义在 ：~\DbEnt\SysEnum\ConfigKey.cs

直接对枚举项取值： ConfigKey.InitPassword.Get()

### C#（服务器）端开发API

见 Common的帮助文档。

### 代码生成器

#### MyOql实体生成器

运行 MyTool 项目的 《生成实体代码.bat》，自动生成实体代码，需要注意的是：

需要有R盘 ， 如果没有，可以使用 DOS 命令创建: subst D:\r r: 其中 d:\r 是一个文件夹

app.config 中配置数据库连接字符串， 修改后并重新编译MyTool 项目，使之更新 bin\MyTool.exe.config

当没有唯一列，没有自增列，主键值是动态生成的（触发器）。需要有如下规则：

* 主键列中的动态列，只能有一个，且需要定义到计算列中。
* 在触发器中生成的主键值，并且最后要显式的 select 出来。

#### 页面生成器

## Web（浏览器）端定义

### 开发标准

1. 所有实体都会生成列表，卡片，添加，修改，删除的功能页面。在该基础上进行开发。

其中列表可以自动表示为引用。

1. FlexiGrid中如果要显示关联数据，要保持前端定义的关联列在主列表。用以支持排序：如主列表是产品表，显示产品类型的Name。那么主列表列定义还是ProductTypeID而非自定义的列属性ProductTypeName。
2. 卡片UI规范：有组标题。在MyJs\_Extend中定义了组标题颜色条（可通过Css定义两种颜色）。
3. Url中不能用以下名字做为参数，因为它们需要在客户端另有用处：

Url中的参数，会生成到客户端的 jv.page() 对象里。

area,controller,action,uid 是Mvc的路由参数，会生成到客户端的 jv.page() 对象里。

当前页面方法定义到 jv.page() 对象上，可在 document.ready 里把 jv.page() 对象缓存下来。如：

$(function(){

var page = jv.page() ;

page.function1 = function(){ } ;

});

1. 所有事件的源头，都要把 event 传递过去，次序不限，也可以不接收。取事件发生源用jv.GetDoer().所有的Ajax事件，都要设置beforeSend事件，在该事件中把XmlHttpRequest对象设置originalEvent和 target 属性，以支持jv.GetDoer，来达到穿透 Ajax的目的。
2. Dom元素命名避免与 浏览器Js 的全局对象，事件名称冲突，在 IE 下有冲突。

### 系统定义的Css Class

* 1. FillHeight表示该元素自动填充所有的可用高度（济览器最底端 - 该元素弟弟元素的高度）。最底端=Max(可见区最底端,Document.Height) ，如果要指定偏移量，需要给对象设置 ofvw （offsetvalueOfwidth） 属性。
  2. FillHeight- 后面多一个减号，表示只在执行window.onload 时才调整该样式的高度。
  3. FillToBottom 伴随 FillHeight 使用，表示FillHeight设置的高度到浏览器底部。
  4. FillWidth 表示该元素自动填充所有的可用宽度（浏览器最右端 – 该元素弟弟元素的宽度）。如果要指定偏移量，需要给对象设置 ofvh( offsetvalueOfheight)属性。
  5. FillWidth- 后面多一个减号，表示只在执行 window.onload 时才调整该样式的高度。

Div 表示 Table 示例代码：

<div class="PmTool PinWidth">

<div class="Inline" style="width: 30%">

<input type="button" value="查询" onclick="jv.page().flexiQuery();" />

<input type="button" value="清空" /></div>

<div class="Inline center" style="width: 30%;"> 中间文本

</div>

<div class="Inline right FillWidth-" >

<input type="button" value="新增" id="addBillType" />

<input type="button" value="打印" onclick="return jv.popPrint();" />

</div>

</div>

* 1. FillToRight 伴随 FillWidth 使用，表示FillWidth设置的宽度到浏览器右端。
  2. pin 钉住元素。用在以下地方：
     1. 钉住MyTool，在 MyTool 里的按钮会根据页面需要移动到 .BoxyTool 或 .PageTool 中去, 定义了 pin 之后, MyTool元素不会移动。
     2. 钉住弹出引用的列表列，在弹出引用页面，列表只显示 bind = true , 和 col.name=”View” 且 col.css=”pin” 的列。其它列不显示。
     3. 钉住弹出引用的列表中的按钮，在弹出引用页面，列表按钮只显示 bclass = “query” 和bclass=”pin” 的按钮。
     4. (可以写作：Pin)钉住 .val 显示内容，显示内容会被显示在 “.val .pin” 上，如果没有”.val .pin” ，则会显示在 “.val”上
  3. PinWidth 固定宽度值，如果要设置偏移量，需要给对象设置 ofvw 属性。
  4. jvRequest ,开发者不要自行定义该样式,它是页面系统标识页面作用域的类.
  5. kv 表示页面 标签和输入框组合类. 标签用 key ,输入框用 val ,样式会设置宽度。

kv模板：

<div class=”kv”>

<span class=”key”>名称：<input type=”hidden” id=”Edit\_Name” /></span>

<span class=”val”><input readonly=”readonly” class=”ref” /></span>

</div>

如果是 Td，则是：

<td class=”key”>名称：<input type=”hidden” id=”Edit\_Name” /></td>

<td class=”val”><input readonly=”readonly” class=”ref” /></td>

* 1. key
  2. val
  3. ref 表示是一个引用，作为同 .val .pin，用于控制显示样式。
  4. jvDetail 表示在卡片页面上，该内容只能是显示的内容。 只显示的内容，弹出的对象是查看卡片
  5. MyDate 弹出jQuery的日期控件
  6. MyTime 弹出 jQuery 的时间控件
  7. MyDateTime 弹出 jQuery 的 日期时间控件
  8. MyCardTitle 卡片标题彩带
  9. MyTool,myTool 卡片工具条。 在Boxy 里，工具栏按钮会移动到 .BoxyTool 中, 在 页面上时,会移动到 .PageTool 里. 如果没有这两个类, 则保留 MyTool 工具条。 其中 .MyTool 是带有样式的。
  10. .MyTool>.view 仅在MyTool的子级有效，表示仅在Detail时显式。SetDetail方法去除非.view的元素。
  11. BoxyTool 定义在Boxy标题栏右侧的 按钮区.
  12. PageTool 定义在 页面 Caption 右侧的 按钮区
  13. MyCard 卡片主体
  14. divSplit，表示卡片中间的分隔线，内容应该为空。
  15. Main 定义在 Master 里的内容主体
  16. Link 模拟链接样式
  17. Inline 把 display=block 的元素 设置为 流式布局.

### 系统函数

* 1. jv.page(con)

作用：取当前Js的作用域。

在 jv.page()的函数定义内部，使用 this 取当前作用域，好处是：避免使用 jv.page() 。 jv.page() 具有不确定性。当在子页调用父页的方法时：

子页：

Jv.page(jv.parentBoxdy()).CommID = 1 ;

Jv.page(jv.parentBoxdy()).flexiQuery() ;

父页：

Jv.page().flexiQuery = function()

{

**var id = Jv.page().CommID; //有问题。下注**

**//正确写法：**

**var id = this.CommID ;**

}

注解：这时的事件源其实还是之前的子页里的元素，所以 jv.page() 是子页。

正确的做法是取当前调用 flexiQuery 的页。 理所当然是使用 this即：

var id = this.CommID ;

* 1. jv.boxdy()
  2. jv.IsNull(obj)
  3. jv.SplitWithReg()
  4. jv.MyGuid()
  5. jv.encode()
  6. jv.decode()
  7. jv.JsonValuer()
  8. jv.HasValue()
  9. jv.CreateEventSource
  10. jv.GetDoer()
  11. jv.LoadJsCss()
  12. jv.Pop()
  13. jv.PopDetail()
  14. jv.PopList()

弹出指定Area，Controller 的List页面。

参数见源码定义。

* 1. jv.PopListConfig()
  2. jv.PopDetailConfig()
  3. jv.ExecHtml()
  4. jv.ExecJs()
  5. jv.url()
  6. jv.UnloadReqeuest()
  7. jv.AddRequest()
  8. jv.RegisteUnload()
  9. jv.TestRole()
  10. jv.IsInBoxy()
  11. jv.Hide()
  12. jv.MyOnInit()
  13. jv.MyOnLoad()
  14. jv.TimeSpan
  15. jv.Upload

### 系统处理的扩展方法

* 1. Array.indexOf
  2. Array.max
  3. Array.min
  4. Array.removeAt
  5. Array.remove
  6. Array.intersect
  7. Array.minus
  8. Array.lmFirst
  9. string.format
  10. string.PadLeft
  11. string.PadRight
  12. string.trimAll
  13. string.trim
  14. string.byteLen
  15. string.getDate
  16. string.isGuid
  17. Array.Recursion

### 扩展的 jQuery函数

* 1. GetDivJson
  2. SetDivJson
  3. IsBindEvent
  4. out
  5. SetDetail
  6. hooked
  7. OneClick
  8. LoadView

### 数据表格（FlexiGrid）：

#### 常规功能

var myGrid = $(".myGrid", jv.boxdy());

myGrid.flexigrid({

title: "",

url: "",

role: { id: "Id", name: "CostName"},

useId: true,

colModel: [

{ display: "Id", name: "Id", width: 0, sortable: false, align: "left", hide: true },

{ display: "", toggle: true, name: "CostName", width: 50, sortable: false, align: "left" }

]

});

更多参数，参见 flexiGrid 源码的参数列表及备注。本文档中 myGrid 是jQuery 对象， grid 是 flexigrid 对象。

列参数列表： 列参数是Json数组，每项Json除 bind 属性外，其它属性，都支持回调。 format 和 html 是运行时回调（有行数据参数），其它是静态回调（即没有参数）

display ： 字符串， 表示显示名称

toggle ： 布尔， 表示是否可切换显示列。

bind ： 布尔， 不支持回调， 表示是否是数据绑定列

name : 字符串 ， 表示唯一name

onlytd ： 布尔， 表示td 内容是否只显示td，如果为false，则用 div 填充。

css ： 字符串，表示列的class

sortable ： 布尔，表示是否可排序，刷新排序

align ：字符串（left,right,center）,表示TBODY中td文本的对齐方式。如果要实现thead对齐，要单独设置

width ：字符串（可设 auto）,数字 ，表示列宽度。 如果为 0 ， 表示隐藏。

format ：function，模板定义，表示单元格内容。

html ：同 format

maxWidth :字符串，数据，表示列的最大宽度

hide : 布尔，表示是否隐藏该列

format: 用于格式化显示方式，有两种方式，

1).$#$ 静态方式： <a href="~/$Id$.aspx" >$#$</a> 用相应的列替换。

2).回调: (row, rowIndex, g, td) .依次传入从服务器返回的当前行数据，行索引，flexigrid对象，当前正在呈现内容的单元格。

flexigrid 参数如下：

* 运行期取flexigrid 对象 myGrid.getFlexi()
* 运行期取flexigrid 参数 myGrid.getFlexi().p
* 页面加载即执行查询，直接调用查询按钮方法： jv.page().flexiQuery();
* 查询标准 写法：

jv.page().flexiQuery = function () {

var queryJDiv = $(".divQuery:last", jv.boxdy(this));

myGrid.getFlexi().doSearch(queryJDiv.GetDivJson());

};

#### 扩展功能

* 根据列名取某行某列值方法grid.getJsonValue(rowIndex, "列名") //grid是FlexiGrid ，rowindex: 行号
* 把从服务器传回的行数据，转换为更方便操作的Json数据：grid.getRowJson(rowData);
* 根据Html元素的行，取得Json数据： grid.getRowJson(jTr, IsGetFromHtml) ;

其中，jTr 可以是 tr 元素，也可以是 tr 的jQuery 对象，也可以是 事件源在该行内的元素。

IsGetFromHtml 表示，是否从Html元素呈现，如果是否，则根据行Id，从缓存数据中转换。

* 如果在服务器端单独执行 LoadFlexiGrid 得到表数据，则需要设置两个Session项：

Session[“FlexiGrid\_id”] 表示客户端定义的表格的 Id列名 。

Session[“FlexiGrid\_cols”] 表示客户端定义的表格的 列名 定义。 采用 逗号 分隔。

RenderJsVar(“table1”,dbr.Menu.LoadFlexigrid( dataset , 200 ) ;

在客户端使用数据源： $(“.myGrid”).getFlexi().p.DataSource = jv.page().table1 ;

* 取选中行：

$(“.myGrid”).getFlexi().getSelectIds();

* 设置全选

可以对 useSelect参数设置多种多个值：

“before”: 在列最前插入默认全选

“after”: 在列最后插入默认全选

数字： 在列指定索引的位置插入默认全选

列定义Json 扩展，可以指定 colIndex，值可以是： “befor”,”after”,数字 。

* 取 Name 的信息

在删除时，需要提示出操作了哪行数据，使用 grid.getNameInfo( rowId ) 方法.

if (Confirm("请确认是否删除该记录, " + myGrid.getFlexi().getNameInfo(sid))) {

}

#### 规范

* 默认：宽度自适应，标题居中值左对齐，不排序，数据绑定，maxWidth是auto，minWidth是auto
* 时间值 宽度 120 ， 左对齐
* 按钮 ，居中对齐
* 固定小于等于两个字的宽度，如按钮设置为 50

### 圆角方案：

* 1. 静态Html圆角：

增加属性 border-radius=”value”.

* 1. 动态添加的Html元素添加圆角：

$(添加的元素).MyCorner({radius:value});

* 1. 皮肤文件定义圆角

在主题文件夹下的SkinVar.js 文件内，定义各个圆角的类名及圆角的值。系统自动加载该文件

* 1. 取消圆角：

$(圆角元素).MyCorner({unCorner:true}) ;

* 1. Value的取值：

一个数字值：表示四个圆角用相同大小的半径做圆角。0值不处理圆角

“，”分隔的四个数值：分别表示左上，左下，右上，右下的四个圆角，与Css3标准相同

* 1. 圆角对Css的约束条件
     1. position= absolute or relative
     2. 去除 overflow 属性。
     3. display = block or inline-block
     4. 尽量不要有 line-height ，在IE7下会显示上下错位。

### TextHelper方案：

服务器端写法：

Html.RegisteEnum("List\_ParkCategory", ParkingCategoryEnum.AllParking)

.Input(new { @class = "dd" })

.Radior();

替代 Html 的select 元素，增强显示效果

用法如下：

$("#Edit\_MySkin\_Display").TextHelper({ post: "click", url: "~/Admin/Dept/GetSkins.aspx", datatype: "array", hidden: "Edit\_MySkin" });

参数说明：

* 1. post：何时Post，可选值：
     1. chang：表示值改变时 Post，自动完成效果。
     2. click：首次单击时Post，自动第一次获取数据。
     3. none：不Post获取数据
  2. url：获取数据的地址。
  3. datatype:数据格式，可选值：
     1. array: 数组
     2. json: Json数据
  4. hidden：动态生成 Hidden 的ID，用于自动Post。见GetDivJson。

### Boxy方案：

Boxy用于弹出遮罩层。用法如下：

* 1. 创建本页面的遮罩层：new Boxy($(Div本页面层对象), { modal: true, title: "查询" }).resize(400, 100)
  2. 关闭：Boxy.get(btnID).hide();
  3. Ajax打开其它页面代码模板：

Boxy.load("~/Admin/Dept/Detail/id.aspx", { filter: “#btnView”, modal: true, title: "查看详细" }, function (bxy) {

bxy.resize(680);

});

Load第一个参数是要Load的Url

第二个参数是一个对象参数如下：

filter:要加载的内容的ID

modal:是否要模态弹出。尽量用true。

title：标题

第三个参数是创建Boxy成功的回调函数，其中resize函数，只需指定宽度，高度会自适应（最大高度是当前窗口的可见高度）。

* 1. 替代 alert和 confirm方案：

Boxy.ask(“确认要进行该操作吗？”, ['OK',’Cancel’], function(res){

if ( res==’OK’){}

else {}

})

* 1. 把另外一个页面当成函数，入口参数是Url里的参数，返回参数需要进行如下设置：

Boxy.load，增加okCallback参数。

被调用页面设置Boxy的okValue参数。

例子：

调用者：

Boxy.load("url", { filter: btnID, modal: true, title: "修T改?" }, function (bxy) {

bxy.resize(580);

},

function (d) { alert(d); });

被调用者：

Boxy.getOne().okValue = function () { return"OK"; };

Boxy规范如下：

1. 一个页面只能有一个弹出Boxy（弹出TextArea或Ask除外），对于其它的弹出Boxy用弹出窗口替代。(弹出窗口参见客户端弹出窗口方案)
2. 对于大文本（TextArea），系统自动弹出Boxy，尽管是在一个Boxy的弹出页面上。

### 客户端弹出窗口方案

弹出列表：jv.PopList(setting);

弹出卡片：jv.PopDetail(setting);

setting 对象格式如下：

obj: 事件对象的源

area: 要弹出的实体所属的 Area（Mvc概念）,

entity: 要弹出实体名称,

callback: 弹出成功后回调函数,

width: 弹出窗体的宽,

height: 弹出窗体的高,

mode: 弹出窗口的模式：radio,check,none,单选，多选，不选。

uid: 弹出窗口的附加参数（Mvc规范）,

query: {}, 弹出窗口的参数，Json格式 .

target: 弹出窗口对象的名称，可省略。

客户端弹出比较复杂，Mvc系统用服务器扩展方法PopForCheck和PopForRadio，PopOnly方法来弹出。服务器端代码如下，以PopForRadio为例：

publicstaticstringPopForRadio(thisHtmlHelper html, string Id, stringDisplayInnerHtml, string Area, ColumnClipEntityId, ColumnClipEntityName, string Code, string Name)

它会生成前面有显示标题（可点击），后面显示值的格式。

Id：生成控件值的 Id

DisplayInnerHtml：用于显示标题

Area：要弹出实体的Area

EntityId：要返回实体的 Id 项。

EntityName：要显示实体的名称项

Code：默认的 Value

Name：默认的显示名称。

**其事件顺序如下：**

1. 查找 hidden 和 textbox ，得到当前引用值，供列表默认选中项。

2. 根据配置得到弹出窗口的大小和位置 （FindPopConfig）

3. 根据参数生成 Url ，并拼接 Url的QueryString 和引用类型（单选还是多选）

4. 根据情况确定 打开模态窗口，还是非模态窗口。（仅在IE下有效，且 1.指定 UseModalDialog 或2非报表URL ）

5. 打开引用

6. 执行 PopList 的 loadCallback

**用户点击 确定， 或单选的双击 后的事件执行过程：**

1. 执行 PopList 参数的 refClick 回调

2. 给控件赋值。

3. 执行 PopList 参数的 callback 回调

4. 触发 hidden 和 textbox 的 change 事件

5. 关闭引用窗体

1. 客户端助手函数

Array对象.IndexOf(Index) ：返回指定数组索引的值。

Array对象.max() ：返回数组中最大的值

Array对象.min() ：返回数组中最小的值

Array对象.insertAt(Index) ：在指定索引位置添加值

Array对象.removeAt(Index) ：移除数组中指定索引的值

String对象.format(args):示例： “{0} {1} !”.format(“hello”,”world”); 返回 hello world !

String对象. trimAll () : 去除所有空格，包括前面，后面，中间的。

简化js技巧：

* + - 1. 连续定义变量使用一个var，如vara,b,c,d;
      2. 如果有如果有消息，则alert后返回，可以这样写：  
         if ( res.msg ) return alert(res.msg) ;  
         如果返回 false , 可以改为：  
         if( res.msg ) return alert(res.msg) || false ;

1. 卡片与服务器端的交互

打包客户端数据：jv.GetDivJson(selector,prefix) 返回页面Json对象.

1. 页面当作嵌入控件，使用jv.LoadView ,但是，使用加载多个控件时，需要：

它时，不能并行加载，如双列表。

1. 在单卡单表结构中应用jv.Request

varreq = jv.AllRequest[jv.AllRequest.length - 2]; //由于jv.LoadView会增加一个jv.Request,所以要取前一个Request

### 弹出查询

场景：当查询条件很多，当前页面只显示主要查询条件，弹出其它查询条件时的方案。

实现目的：只是UI的展现方式不同，实则开发人员可认为，所有查询条件都在当前页面。

实现原理：把当前页面上额外的查询条件的Html内容（仅是Html内容）转移到弹出窗口，待用户输入查询条件后，再同步回当前页面。

实现方案：

在popSearch 容器中，定义弹出的查询条件，如果默认要隐藏某一行，要在该行上显式定义是否可见

把要执行的脚本也放到 “.popSearch” 容器中。

示例：

<table class="popSearch" >

<tr>

<td class="key">

名称

</td>

<td class="val">

<input type="text" id="Pop\_Name" value="li"/>

</td>

</tr>

<tr style ="display:none" >

<td class="key">

排序

</td>

<td class="val">

<input type="text" id="Pop\_Sort" />

</td>

</tr>

<tr style ="display:none"><td>

<script type="text/javascript">

$(function () {

$("#List\_Name").TextHelper({ data: { zhang: "张", li: "李"} });

});

</script>

</td></tr>

</table>

<input type="button" id="retrieval" class="button" onclick="jv.popSearch('#doQuery')" value="检索" />

jv.popSearch 的参数：

querySelector: 查询按钮的选择器，在设置完检索条件后，自动触发查询。如果为空，则不触发查询。

### 验证

用法：

验证属性有 chk , chk\_msg , chk\_val 三个属性

* + - 1. chk 表示验证规则 。
      2. chk\_msg 表示当验证不通过时，显示的消息。
      3. chk\_val 表示要验证值的控件。可以是：回调，jQuery选择器。

当验证控件的值不符合验证规则时，显示错误消息。

chk用法如下：

* + - 1. 函数式验证

有两个参数，分别是 val，msg。 表示 chk\_val的值和chk\_msg 的值。

chk=”function(val,msg){ if ( val.length == 0 ) return msg ; }”

* + - 1. 简化函数式验证

是对函数式验证的简写，以“：” 开始，省略 functon(val,msg){ } 这部分，只写里面的内容。如：

chk=”: if ( val.length == 0 ) return msg ;”

也可以在函数内部进行额外的验证：

chk=”: if ( jv.chk\_one({chk:’datetime’,chk\_val:’#startTime’}) == false ) return ‘请输入时间’ ; return ;

* + - 1. 自定义表达式

与正则相似，但不相同。

格式： 类型(开始值，结束值){值最小长度，值最大长度}

类型有： int,number,date,datetime,time,email,enum

* + - 1. 正则表达式

chk=”reg == /:d+/”

chk=”reg != /:d+/”

### 富文本编辑器

系统采用 tiny\_mce，并对其进行了部分改造（中文字体大小及附件）

在页面引入两个Js文件：

<script src="~/Res/tiny\_mce/jquery.tinymce.js" type="text/javascript"></script>

<script src="~/Res/tiny\_mce/Mytinymce.js" type="text/javascript"></script>

对textArea 进行富文本化：

$("#content").Mytinymce();

### Js 的精度问题

由于Js的精度问题，从服务器端返回到客户端时，不要返回大于 Int，Float数据类型的Number类型数据。如 long型，double型。可以用 string 类型来代替。

Ie的四舍五入，有Bug：<http://www.cnblogs.com/MythYsJh/archive/2009/02/13/1389843.html> ，而 Math.round函数是银行家舍入（四舍六入）：<http://yyg329-405.blog.163.com/blog/static/3780926620117895954262/>

“四舍六入法”可以概括为：“四舍六入五考虑，五后非零就进一，五后皆零看奇偶，五前为偶应舍去，五前为奇要进一。”

我们系统采用 Number.toFixed 的修正版。详见 myJv.js

### 汉字转化为拼音

方法一:

添加Pinyin.js的引用

CC2PY（str）汉字转化成拼音,输入汉字转化成拼音，字母和数字原样输出，其他字符过滤:

PinYinCapLetters(str): 汉字转化成拼音并且只取首字母,输入汉字转化成拼音，字母和数字原样输出，其他字符过滤

示例如下:

<script src="Pinyin.js" type="text/javascript"></script>

<script type="text/javascript">

document.write("汉字转化成拼音,输入汉字转化成拼音，字母和数字原样输出，其他字符过

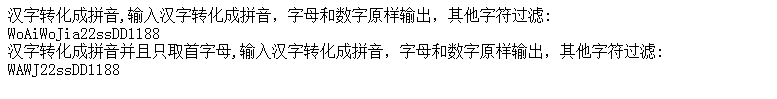
滤:<br/>"+CC2PY("我爱我家22ssDD11##@@88")+"<br/>");

document.write("汉字转化成拼音并且只取首字母,输入汉字转化成拼音，字母和数字原样输出，其他

字符过滤:<br/>" + PinYinCapLetters("我爱我家22ssDD11##@@88") + "<br/>");

</script>

运行效果如下:



方法二:

脚本:

<script src="pinyinW.dict.src.js" type="text/javascript"></script>

<script src="pinyinW.js" type="text/javascript"></script>

<script type="text/javascript">

function tt(o) {

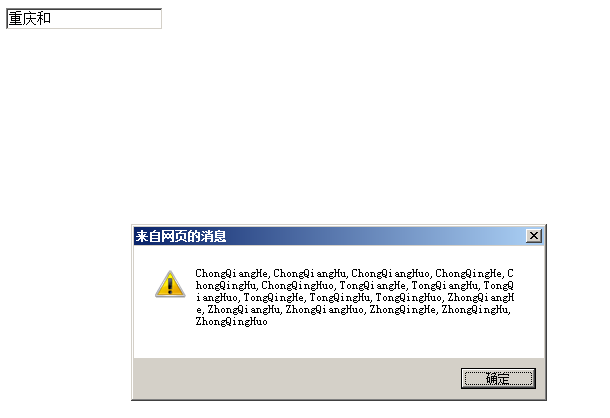
alert(pinyin(o.value, false, ","));

}

</script>

<input type="text" id="test" onblur="tt(this)"/>

效果:



返回的是所有汉字拼音组合的可能性

## MyOql

MyOql是一个ORM组件，它具有如下特点：

* + - 1. 可以对表，视图，存储过程，表值函数进行映射。
      2. 可以扩展自定义聚合函数。
      3. 可以对实体分组。

在产品设计前期，对产品划分的模块，在MyOql中对应为数据实体的“组”（目前只能有一级组）。

同级下只能有一个组名为空。

* + - 1. 基于配置文件的代码生成工具。
      2. 支持众多SQL关键字，参数化查询。
      3. 集成权限，日志，事件过滤。

### 使用说明

建库，建表，编写 MyTool 项目中的 app.config 配置文件，在实体层定义数据库枚举项，使用 《生成实体代码.bat》 生成实体到 实体层，实体层引入MyCmn,MyOql ， 即可使用。

可生成如下内容：

* + - 1. 数据库表，继承自 RuleBase, ITableRule
      2. 视图， 继承自 RuleBase,IViewRule
      3. 表值函数，继承自 RuleBase,IFunctionRule ， 生成为 dbr 下的方法
      4. 存储过程，生成为 dbr 下的方法
      5. 生成针对实体主键的 FindBy单主键 和 DeleteBy单主键 方法

### 数据库名称与程序名称映射机制

强制把数据库名称映射为程序名称，如下查询：

dbr.Menu.Select(o=>o.Name.As(“name”)).ToMyOqlSet();

返回的列名是 “Name”

其规则如下：

如果数据库名称中出现以下 指定的分隔字符 , 则按以下分隔字符 进行分隔, 并把每个部分首字母大写之后进行联接. 分隔字符有:

1.空格

2.下划线 \_

3.加号 +

4.减号 -

5.#

6.&amp;

7.竖线 |

8.冒号 :

9.分号 ;

10.小于号 &lt;

11.大于号 &gt;

12.逗号 ,

13.点号 .

14.$

15.左括号

16.右括号

如果没有出现上述分隔符, 如果数据库名称 全大写,或全小写, 则按首字母大写转换, 否则 返回数据库名称.

### 功能示例

实体入口 dbr,各例实体不确定。

#### 查询

分页

dbr.Menu

.SelectWhere()

.Skip( 30 )

.Take( 15 )

.ToEntityList(o=>o.\_) ;

左,右连接

dbr.City

.Select(o=> new Columns( o.Id.As(“CityId” , o.Name.As(“CityName”)))

.LeftJoin( dbr.State , (a,b)=> a.StateId = b.Id) , o=>o.Name.As(“StateName”) )

.ToMyOqlSet() ;

分组及分组过过滤

dbr.Menu

.Select(o=>new Columns(o.Name , o.Count().As(“Count”)))

.AutoGroup()

.Having(o=>o.Count() > 1 )

.ToMyOqlSet();

排序

dbr.Menu

.SelectWhere(o=>o.Id < 100 )

.OrderBy(o=> o.Name.Asc)

.OrderBy(l=>o.Url.Asc)

.ToMyOqlSet();

组合

dbr.Menu.SelectWhere(o=>o.Id < 50 )

.Union( dbr.Menu.SelectWhere( o=>o.Id > 100 ) )

.ToMyOqlSet() ;

子查询

dbr.Menu

.Select(o=>new Columns(o.Name , o.Count().As(“Count”)))

.AutoGroup()

.Having(o=>o.Count() > 1 )

.SelectWrap(“a”)

.Join( dbr.Menu , (a,b)=> dbr.Menu.Id.FromTable(“a”) == b.Id )

.ToMyOqlSet();

#### 添加

添加单条

dbr.Menu.Insert(o=>o.Name == “菜单根” & o.Url = “~/DeskTop.aspx” ).Execute() ;

Insert … Select …

dbr.Menu.Insert(dbr.PmMenu.Select(o=>o.Id > 100 )).Execute() ;

获取插入后的自增Id 方式1：

var insertMenuClip = dbr.Menu.Insert(o=> o.Name == “菜单根” & o.Url = “~/DeskTop.aspx” );

var affectRow = insertMenuClip.Execute() ;

var dbAutoId = insertMenuClip.LastAutoID ;

获取插入后的自增Id 方式2：

操作的实体自动返回自增列及计算列内容。

var ent = new MenuRule.Entity() ;

ent.Name = “菜单根”;

ent.Url = “~/Desktop.aspx”;

dbr.Menu.Insert( ent ) .Execute() ;

var dbAutoId = ent.Id ;

#### 修改

追加列更新

dbr.Menu.Update(o=>o.Name == “1”).Set(o=>o.Url == “~/DeskTop.aspx”).Execute();

重新设置更新列：

dbr.Menu.Update(o=>o.Name == “1”).SetColumns(o=>o.Text == “1”).Execute();

多表更新

dbr.Menu

.Update(o=>o.Name == dbr.PmMenu.Name)

.Join( dbr.PmMenu , (a,b)=>a.Id == b.Id )

.Where ( o => o.Id > 10 )

.Execute() ;

#### 删除

dbr.Menu.Delete(o=>o.Id.In( dbr.PmMenu.Select( m=>m.Id ) .Where (m=>m.Id < 100 ) ) ).Execute() ;

#### 事务

优先使用 TransactionScope （分布式事务） ，代码整洁

using (var tran = new TransactionScope())

{

…

tran.Complete();

}

指定事务（单库事务）

using( var conn = dbr.Menu.GetConnection() )

{

conn.Open() ;

using ( var tran = conn.BeginTransaction())

{

dbr.Menu.Insert( menuEntity ) . UseTrnsaction( tran ).Execute() ;

dbr.Log.Insert( logEntity ) .UseTransaction( tran ) .Execute() ;

try

{

tran.Commit() ;

}catch(Exception e )

{

tran.RollBack() ;

}

}

}

#### 从Menu表中得到无参数的URL

/\*得到无参Url

select substring( Url , 0, case when ind = 0 then 4096 else ind end ) as Url , id,[Text] from

(

select charindex( '?' ,Url ) as ind , \*

from Menu

) as m

where charindex( '/List.aspx',Url ) > 0

\*/

dbr.Menu.Select(o => new Columns(o.Id, o.Url.StringIndex("?").As("UrlIndex")))

.SelectWrap("m",

dbr.Menu.Id.FromTable("m"),

dbr.Menu.Url.SubString(new ConstColumn(0),

dbo.CaseWhen(new ConstColumn("UrlIndex").FromTable("m") == 0, new ConstColumn(4096))

.ElseEnd(new ConstColumn("UrlIndex").FromTable("m"))

).As("Url")

)

.Where(new ConstColumn("Url").FromTable("m").StringIndex(url) > 0)

.ToMyOqlSet();

### 支持的系统函数及关键字

见： SqlOperator ， SqlKeyword 两个枚举。

## 解决方案

### 权限方案: Udi

1.因为给FlexiGrid定义了List，而它的 Query 做为获取数据的Action，是所有动作都可以访问的，所以，PowerAction中就不必再定义 Query ！

2.只有SysAdmin登录的时候拥有最大权限。

3.如果要定义View和Edit，要在PowerColumn中定义View和Edit，并在权限中分配它。所以这里要尽量避免在实体上定义 View 和 Edit 列。解决冲突的方法是前后端修改绑定列名为Table.Column格式的全称。

1. 列表调用

如果默认列表有数据,那么,只需在flexigrid上绑定:

onSuccess: function (grid) { jv.TestRole({ grid: myGrid.getFlexi() }); }

如果默认列表没有数据,而需要按钮查询触发,那么, 除上述添加onSuccess外,还需要在flexiGrid函数后, 调用:

jv.TestRole({ grid: myGrid.getFlexi() });

1. 卡片调用

依赖客户端变量jv.page().entity , 有两种方式能够设置该变量, 一是在前台设置:

jv.page().entity = “TUser,TRole” ;

另一种方式是在服务器端的 Action 设置

RenderJsVar(“entity”,”TUser,TRole”) ;

然后需要调用jv.TestRole () ;

### 排序方案

场景：列表中记录排序。用户可以拖动表的某一行进行当前页的排序，也可以多选行进行 置顶，置底 操作。数据库表用 SortID （Int）存储排序号。

特点：

1. SortID 的初始值是行记录的自增ID 。在拖动的过程中，也要保证SortID 是唯一的。
2. 拖动行时，需要在 flxeiGrid 的 onSuccess 事件进行如下设置：

onSuccess: function (grid) {

jv.ReSortFlexi({ table: grid, SortColumn: "SortID", callback: function (startSortID, endSortID) {

} });

}

1. SortID 按从小大顺序排列。
2. 为了保证SortID唯一性和数据完整性，采用表列锁机制。

算法：

在每次查询数据列表时，会在当前页面上设置当前列表的 VerID 可以用 jv.page().VerID 来取得，在更新行排序ID时，会调用 PSetTableLock，进行加锁，如果加锁失败，则说明数据已脏，需要重新刷新数据。更新完成后，再调用 PReleaseTableLock 进行锁释放。

#### 单次拖动排序算法：

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 名称 | 说明 | 排序ID |
| 1 | Table | 表名 | 1 |
| 2 | Column | 列名 | 2 |
| 3 | VerID | 版本号，每次版本号自增1 | 3  beginSortID |
| 4 | User | 使用者，每次操作，生成GUID | 4 |
| 5 | .。。 | 。。 | 5 |

如：把第4行移动到第一行。变为：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 名称 | 说明 | 排序ID | 新序  endSortID |
| 4 | User | 使用者，每次操作，生成GUID | **4** | **1** |
| 1 | Table | 表名  RowId | 1 | 2 |
| 2 | Column | 列名 | 2 | 3 |
| 3 | VerID | 版本号，每次版本号自增1 | 3 | 4 |
| 5 | 。。。 | 。。。 | 5 | 5 |

客户端向服务器回传改变后的 两个排序ID： beginSortID（4），endSortID（1），即由第4条变成第一条。由于SortID一直是顺序的，所以需要如下过程：

1. 先把beginSortID 的SortID设置为Null。并记录下该条记录的ID，标识为 rowId
2. 把 小于beginSortID 到 大于endSortID 之间的（不包含），全部+1
3. 把rowId 记录的SortID 更新为 endSortID

如把第1行移动到第四行。变为：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 名称 | 说明  RowId | 排序ID | 新序  beginSortID |
| 2 | Column | 列名 | 2 | 1 |
| 3 | VerID | 版本号，每次版本号自增1 | 3 | 2 |
| 4 | User | 使用者，每次操作，生成GUID | 4 | 3  endSortID |
| 1 | Table | 表名 | 1 | 4 |
| 5 | .。。 | 。。。 | 5 | 5 |

相应算法为：

1. 先把beginSortID 的SortID设置为Null。并记录下该条记录的ID，标识为 rowId
2. 把 大于beginSortID 到 小于endSortID 之间的（不包含），全部 -1
3. 把rowId 记录的SortID 更新为 endSortID

综合上述两个算法如下：

1. 先把beginSortID 的SortID设置为Null。并记录下该条记录的ID，标识为 rowId
2. 把 小于beginSortID 到 大于endSortID 之间的（不包含），全部+1
3. 把 大于beginSortID 到 小于endSortID 之间的（不包含），全部 -1
4. 把rowId 记录的SortID 更新为 endSortID

#### 多条置顶置底算法

如把 第2，5条置顶变为：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 名称 | 说明  RowId | 排序ID | 新序  多选序列 |
| 2 | Column | 列名 | 2 | 1 |
| 5 | .。。 | 。。。 | 5 | 2 |
| 1 | Table | 表名 | 1 | 3 |
| 3 | VerID | 版本号，每次版本号自增1 | 3 | 4 |
| 4 | User | 使用者，每次操作，生成GUID | 4 | 5 |

由于按之前单条拖动方案，第一条会被Update两次（第2 条更新一次，第5条更新一次）。所以有采用分段更新。

* + - 1. 先对多选序列分隔，并倒排。 5-2 （不包含），2-0（不包含）
      2. 把多选序列SortID更新为Null，并记录下倒序相应的RowId
      3. 对5-2区间部分的SortID，进行 +1 处理。
      4. 对 2-0 区间部分的SortID 进行 +2 处理。
      5. 把 多选的倒序选择序列，按顺序更新SortID，即 RowId 是 5，2 的记录的SortID 更新为 1，2

如把 第2，5条置底变为：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 名称 | 说明  RowId | 排序ID | 新序  多选序列 |
| 2 | Column | 列名 | 2 | 1 |
| 5 | .。。 | 。。。 | 5 | 2 |
| 1 | Table | 表名 | 1 | 3 |
| 3 | VerID | 版本号，每次版本号自增1 | 3 | 4 |
| 4 | User | 使用者，每次操作，生成GUID | 4 | 5 |

### 表锁

场景：当两个用户同时打开页面， 并更新后， 第一次更新操作会被覆盖。

#### 表列锁

场景：当两个用户同时打开列表页面， 并对多地记录进行更新后（如排序）， 第一次更新操作会被覆盖。

定义表结构：TableLock

|  |  |  |
| --- | --- | --- |
| 列名 | 类型 | 说明 |
| Table | Varchar(50)，主键 | 表名 |
| Column | Varchar(50)，主键 | 列名 |
| VerID | Int | 版本号，每次版本号自增1 |
| User | Varchar(64) | 使用者，每次操作，生成GUID |

实现机制：

在查询列表时，先从TableLock中查询版本号，在对列表进行批量更新，先生成GUID做为使用者，结合查询时的版本号 进行如下判断：

当使用者为不为空时， 表示有用户正在使用。此时不能对该表该列进行更新。

当更新操作所持有的版本号与表列锁中的 VerID 不符时，说明用户所持数据已过期，为脏数据，也不能更新。

如果上述检查通过，则先进行占位处理，即设置User 为该使用者，再进行业务表更新处理，处理完成后， 设置版本号+1， 并清空使用者列。

为简化操作，可使用两个存储过程辅助开发：

PSetTableLock(Table,Column,VerID,User) ，表示先对User进行占位，成功返回 1 ， 失败返回 0

PReleaseTableLock(Table,Column)，表示释放表列锁。 成功返回1，失败返回0

#### 表行锁

场景：当两个用户同时打开卡片页面， 并更新该行记录后， 第一次更新操作会被覆盖。

定义表结构：RowLock

|  |  |  |
| --- | --- | --- |
| 列名 | 类型 | 说明 |
| Table | Varchar(50)，主键 | 表名 |
| RowID | Varchar(150)，主键 | 行主键值 |
| VerID | Int | 版本号，每次版本号自增1 |
| User | Varchar(64) | 使用者，每次操作，生成GUID |

实现机制：

在查询卡片时，先从TableLock中查询版本号，在对卡片进行更新，先生成GUID做为使用者，结合查询时的版本号 进行如下判断：

当使用者为不为空时， 表示有用户正在使用。此时不能对该表该列进行更新。

当更新操作所持有的版本号与表列锁中的 VerID 不符时，说明用户所持数据已过期，为脏数据，也不能更新。

如果上述检查通过，则先进行占位处理，即设置User 为该使用者，再进行业务表更新处理，处理完成后， 设置版本号+1， 并清空使用者列。

为简化操作，可使用两个存储过程辅助开发：

PSetRowLock(Table,RowID,VerID,User) ，表示先对User进行占位，成功返回 1 ， 失败返回 0

PReleaseRowLock(Table,RowID)，表示释放表列锁。 成功返回1，失败返回0

### 批量数据推送

场景：当有多对多的数据关系发生变更时，需要批量把数据推送到数据库，如下例：

用户与角色的多对多关系，在用户卡片上修改了多个角色后，需要把角色推送到数据库，假设之前的数据关系如下：（OriList）

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 用户 | 角色 | 角色ID |
| 1 | 张三 | 香港区管理员 | 1 |
| 2 | 张三 | 北京区管理员 | 2 |
| 3 | 张三 | 天津区管理员 | 3 |

用户更改之后角色为：（NewList）

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 用户 | 角色 | 角色ID |
| 1 | 张三 | 香港区管理员 | 1 |
| 2 | 张三 | 北京区管理员 | 2 |
| 3 | 张三 | 上海管理员 | 4 |
| 4 | 张三 | 福建区管理员 | 5 |

可能是情况只有两种： 一种是 新增数据，另一种是删除数据。

#### 新增数据

新增数据集= NewList – OriList 。 可作批量插入。BulkInsert

#### 删除数据

删除数据集 = OriList – NewList 。 可以批量删除：Delete Table Where Id in ( DelArray ) ;

### 接口开发

接口采用提供视图供第三方读取的方式。

注意：

在程序之外，即脱离我们的Mvc程序，在修改了表数据后（Update，Delete），要执行PClearAppCache(tab)

PclearAppCache应用通过数据库脚本，第三方应用程序改变了我们的数据库后，通知我们的C#程序清除缓存。

### 事务

执行原则：先执行主表，再执行子表。 没有主子关系，按业务流程先后关系。

分布式事务:见Chm帮助

using (var tran = new TransactionScope())

{

//do something 1.

//do something 2.

try

{

tran.Complete();

}catch( Exception e ){

//自动回滚。

}

}

分布式事务要求每台服务器都要启用 MSDTC，详细设置：

<http://www.cnblogs.com/newsea/archive/2012/06/08/2541280.html>

单库事务:

var conn = dbr.TTask.GetDbConnection();

using (vartran = conn.BeginTransaction())

{

try

{

dbr.Person.Update()

.UseConnection(conn)

.Execute() ;

dbr.Dept.Insert()

.UseConnection(conn)

.Execute();

conn.Commit();

}

catch

{

conn.RollBack();

}

}

### 报表开发

报表分页方案：

<http://www.cnblogs.com/newsea/archive/2012/03/04/2379250.html>

报表样式设置：

<http://www.cnblogs.com/newsea/archive/2012/03/04/2379274.html>

文件结构见《代码规范-文件结构》

如果展现的表格数据源和报表数据源相同，则Biz返回的数据源可以重用。重用原则：

* 返回的数据列是最大集合。
* 自定义查询，需要手动修改传递的 QueryModel (QueryBase)

### 系统服务

见接口方案

### 数据导出

#### 导出数据到Excel2003

参考:<http://www.cnblogs.com/newsea/archive/2011/06/12/2078864.html>

string connString = @"Provider=Microsoft.Jet.OLEDB.4.0;Data Source=R:\MyWeb.xls;Extended Properties=Excel 8.0;";

using (OleDbConnection con = new OleDbConnection(connString))

{

con.Open();

string strSql = "insert into 表 ( 列1,列2 ) values ( ?,? ) " ;

OleDbCommand cmd = new OleDbCommand(strSql, con);

{

var p = new OleDbParameter("?", OleDbType.Integer);

p.Value = 0;

cmd.Parameters.Add(p);

}

{

var p = newOleDbParameter("?", OleDbType.Integer);

p.Value = 0;

cmd.Parameters.Add(p);

}

cmd.ExecuteNoQuery() ;

}

#### 导出数据到Excel2007+(待更新)

var ox = new OpenXml(new System.IO.FileStream(Server.MapPath("~/Doc/blank.xlsx"), FileMode.Open), Server.MapPath("~/Upload"));

ox.LoadWithColumn(dt, "Sheet1", objCol);

return new DownloadResult("出票详情.xlsx", ox.ToExcel());

### 异步加载树列表

1. 设置flexiGrid参数ajaxUrl参数如:

ajaxUrl: "~/cs/Common/BuildingQuery.aspx?pid={0}",

其中 {0} 是该行的ID .

1. 服务器调用LoadFlexiAjaxGrid函数来完成.
2. 参见:~/cs/Common/BuildingList.aspx

### 跨页面传值（按查询的导出方案）

场景：按查询条件导出到Excel。

查询条件比较多，且查询内容也可能比较多，如果使用URL传值的话，可能存在URL超长截断的可能。

实现原理遇到的问题：

在回调（Ajax或timer）里执行 document.location 或 window.open 方法，浏览器会阻止下载（有提示）。要解决如何把条件 Post 到服务器端再执行 window.location 方法而不报阻止 是问题的关键。

原理：

模拟Form本身的post，设置一个不同的 action，在回发之后执行Js回调。

实现：

var postJson;

jv.postOpen('~/Host/Home/Export.aspx', postJson);

### 模拟Session方案

对于使用 HttpWebRequest 进行请求，且需要Session 时，可以按下述方案进行：

1. 客户端使用 HttpWebRequest 进行登录模拟，返回用户名和登录时间的加密令牌（双向加解密），同时设置Session。

2. 客户端再使用 HttpWebRequest 交互的时候，把 加密令牌 放到 Cookie 里。服务器端做解密验证用户名和时间差（不能超过一个小时），验证不通过，返回错误，验证通过，更新Session 时间值。

### 动态创建Grid列

前台应用：

Html：

将原来 <table class="myGrid"></table> 表记替换成<span id="splitSpan"></span>

Js： 先将要显示的列的名称 构建出来

//收款票据

var FeesReceiptsColModel = [

{ display: "", name: "ReceID", width: "0", align: "left" },

{ display: "票据号码", name: "BillsSign", align: "left" },

{ display: "客户名称", name: "CustName", align: "left" },

{ display: "<input type='checkbox' id='checkAll'>[全选]", name: "Type", width: "50px", sortable: false, align: "center", html: "<input type='checkbox' value='$ReceID$'/>" }

];

//预交收款票据

var QueryPreCostsDetailModel = [

{ display: "", name: "RecdID", width: "0", align: "left" },

{ display: "", name: "ReceID", width: "0", align: "left" },

{ display: "客户名称", name: "CustName", align: "left" },

{ display: "房屋编号", name: "RoomSign", align: "left" }, { display: "<input type='checkbox' id='checkAll'>[全选]", name: "Type", width: "50px", sortable: false, align: "center", html: "<input type='checkbox' value='$ReceID$'/>" }

];

//触发Grid列的改变

$("#List\_DrReceive").change(function () {

switch ($(this).val()) {

case "1":

jv.page().GenTable(FeesReceiptsColModel, "~/cost/BillIssueBrowse/QueryBillFeesReceipts.aspx");

break;

case "2":

jv.page().GenTable(QueryPreCostsDetailModel, "~/cost/BillIssueBrowse/QueryPreCostsDetail.aspx");

break;

}

jv.page().flexiQuery();

});

var myGrid;//声明全局变量存放Grid对象

jv.page().GenTable = function (colModel, url) {

if ($(".myGrid").length) {

$("#splitSpan").next().remove(); //移除生成的html代码

};

var tab = document.createElement("table");

$("#splitSpan").after(tab);

myGrid = $(tab).addClass("myGrid");//重新构造Grid对象 并且赋给myGrid全局变量

myGrid.flexigrid({

title: "",

url: url,

colModel: colModel,

dataType: "json",

buttons: [],

usepager: true,

rp: 15,

ShowTableToggleBtn: true

});

};

jv.page().flexiQuery = function (ev) {

var searchOption = $(".divQuery:last", jv.boxdy()).GetDivJson("List\_");

myGrid.getFlexi().doSearch(searchOption);

};

### 文件上传

前台应用：

Html:

<tr>

<td class="key">

文件上传：

</td>

<td colspan="3">

<div id="upFile">

</div>

</td>

</tr>

js:

1，声明

$("#upFile").upFile(

{

max: 8,

data: jv.page().fileData,

messagesmaxFile: "最多上传8张照片"

});

2，获取上传的文件id

var ids = $("input[name=upFile]").val();

eg.: ‘1111,2222’

如果有历史上传 : ‘Old\_20120000000001,1111’

带old的是业务附件表的字段

后台代码：

1，显示以前上传的文件：

var files =

"[" + string.Join(",",

dbr.Master.TmRoomAdjunct

.SelectWhere(o => o.RoomID == room.RoomID).ToEntityList(o => o.\_)

.Select(o => new FileJsonResult

{

id = "Old\_" + o.AdjunctCode,

name = o.AdjunctName,

size = o.FileSize.ToString(),

url = ((o.FilPath.AsString().EndsWith("/") ? o.FilPath.AsString() : (o.FilPath.AsString() + "/")) + o.AdjunctCode + o.FileExName)

}.ToString())

) + "]";

RenderJsVar("fileData", new VarJsResult(files) { Type = JsType.Value });

Ps：old前缀用来区分新上传和旧文件

2,保存：

if (Files.HasValue())

{

var fileArr = Files.Split(",");

var notDelList = fileArr.Where(o => o.Contains("Old\_")); //获取不需要删除的附件

if (notDelList.Count() > 0) //删除附件

{

var newlist = Array.ConvertAll(notDelList.ToArray(), o => o.Replace("Old\_", ""));

dbr.Master.TmRoomAdjunct.Delete(o => o.RoomID == Model.RoomID & o.AdjunctCode.NotIn(newlist.ToArray())).Execute();

}

fileArr.All(o =>

{

if (o.HasValue())

{

if (o.Contains("Old\_"))

{

}

else

{

var curFile = dbr.Annex.FindById(o.AsInt());

TmRoomAdjunctRule.Entity adjunct = new TmRoomAdjunctRule.Entity();

adjunct.AdjunctCode = curFile.PmReferenceID;

adjunct.RoomID = Model.RoomID;

adjunct.AdjunctName = curFile.Name;

adjunct.FilPath = curFile.Path;

adjunct.FileExName = curFile.Ext;

adjunct.FileSize = curFile.Size.AsDecimal();

var adres = dbr.Master.TmRoomAdjunct.Insert(adjunct).Execute();

}

}

return true;

});

}

else

{

dbr.Master.TmRoomAdjunct.Delete(o => o.RoomID == Model.RoomID).Execute();//删除全部附件

}

## 数据库系统定义

### 注册CLR程序集

为了更灵活的在数据库端进行逻辑运算，可以把 .Net 程序集注册到SqlServer。

参考：MyClr/说明.sql

|  |  |  |
| --- | --- | --- |
| 表定义 Annex 附件表 | | |
| 列名 | 类型 | 说明 |
| Column | Varchar(50)，主键 | 列名 |

### 视图定义

|  |  |  |
| --- | --- | --- |
| Annex 附件表 | | |
| 列名 | 类型 | 说明 |
| Column | Varchar(50)，主键 | 列名 |

### 函数定义（不包含程序集定义的函数）

|  |  |  |
| --- | --- | --- |
| Split 把字符串拆分成表记录 | | |
| 参数名 | 类型 | 说明 |
| @CharSource | nvarchar(4000) | 要分割的字符串 |
| @SplitChar | Varchar(10) | 分割字符串 |
| GetCount 获取指定字符串的个数 | | |
| 参数名 | 类型 | 说明 |
| @SourceSql | nvarchar(4000) | 源字符串 |
| @find | Varchar(50) | 查找字符串 |
| GetIndexString 取指定位置的 用“，“分隔的记录 | | |
| 参数名 | 类型 | 说明 |
| @SourceSql | nvarchar(4000) | 要分割的字符串 |
| @index | int | 索引号 |
| GetIndexStringSimple取指定位置的分隔的记录 | | |
| 参数名 | 类型 | 说明 |
| @SourceSql | nvarchar(4000) | 要分割的字符串 |
| @StrSeperate | Varchar(10) | 分割字符串 |
| @index | Int | 索引号 |
| GetStringIndex 取某字符串在源字符串中的索引 | | |
| 参数名 | 类型 | 说明 |
| @SourceSql | Text | 要分割的字符串 |
| @StrSeperate | Varchar(10) | 分割字符串 |
| @strFind | Varchar(4000) | 查找的字符串 |
| GetStringIndexSimple取某字符串在源字符串中的索引 | | |
| 参数名 | 类型 | 说明 |
| @SourceSql | nvarchar(4000) | 要分割的字符串 |
| @StrSeperate | Varchar(10) | 分割字符串 |
| @strFind | Varchar(4000) | 查找的字符串 |

### 存储过程定义

|  |  |  |
| --- | --- | --- |
| PLogin 登录验证，返回登录用户的Person 记录 | | |
| 参数名 | 类型 | 说明 |
| @UserID | Varchar(50) | 用户名 |
| @Password | Varchar(50) | 密码 |
| PEditPassword 修改密码，返回影响行数 | | |
| 参数名 | 类型 | 说明 |
| @UserID | Varchar(50) | 用户名 |
| @Password | Varchar(50) | 新密码 |
| PClearAppCache清除程序缓存事件通知，返回影响行数 | | |
| 参数名 | 类型 | 说明 |
| @Tab | Varchar(50) | 数据改变的表名，没有Owner |
| P\_Page根据SQL进行分页返回，返回指定页码的数据记录 | | |
| 参数名 | 类型 | 说明 |
| @strsql | nvarchar(4000) | 没有分页的原始SQL |
| @currentpage | Int | 当前页码 |
| @pagesize | Int | 每页显示条数 |
| Proc\_AllEntity 生成原始的 MyOql实体生成器配置文档 | | |
| 辅助存储过程 | 类型 | 说明 |
| Proc\_GetFKByTableName |  |  |
| Proc\_GetPKs |  |  |
| Proc\_GetPKs |  |  |
| Proc\_AllEntity 生成原始的 MyOql实体生成器配置文档 | | |
| 辅助存储过程 | 类型 | 说明 |
| Proc\_GetFKByTableName |  |  |
| Proc\_GetPKs |  |  |
| Proc\_GetPKs |  |  |

### 程序集定义

|  |  |  |
| --- | --- | --- |
| MyClr 程序集定义了关于 MyBigInt 一些需要.Net的操作。 | | |
| 函数名 | 类型 | 说明 |
| CLRFBitContains |  | MyBigInt是否包含某行 |
| CLRFBitAnd |  | 两个MyBigInt 进行位与操作 |
| CLRFBitOr |  | 两个MyBigInt 进行位或操作 |
| CLRFToPositions |  | 取MyBigInt各个位值 |
|  |  |  |

### 影子库方案

特点：

1. 正常删除业务数据库数据，业务数据库无垃圾数据。
2. 对删除的数据进行转移，转移在删除的触发器中进行。
3. 无额外的代码修改。
4. 影子库和业务库在相同的服务器上，影子库无压力，触发器速度快
5. 每次修改数据库结果，都要重新执行创建过程。
6. 根据业务使用情况，对无需备份的数据，忽略影子库（需要修改过程脚本），目前忽略的有工作流，Log表。

过程：

1. 如果之前存在影子库，先手动删除。
2. 选择正确的业务库。
3. 创建影子库，执行Dev\_CreateShdow4Delete
4. 创建业务库触发器，执行 Dev\_CreateTrigger4Table

### 动态SQL方案

#### 动态查询条件

If 条件1 查询1

else 查询2

转为一句SQL：

( **非**条件1 **or** 查询1 )

**and**

(条件1 **or** 查询2 )

#### RowNumber分页模板

无排序列：

按列排序:

Select \* From

( Select Row\_Number() Over (Order By [\_\_IgNoRe\_\_SubId] asc ) As [\_\_IgNoRe\_\_AutoId], \* From

( Select 1 as [\_\_IgNoRe\_\_SubId] , \* From

(

--自定义SQL，没有OrderBy

) As [\_SubQuery\_]

) As [\_\_SubQuery\_\_]

) As [\_\_\_SubQuery\_\_\_]

where [\_\_IgNoRe\_\_AutoId] between ((@PageIndex - 1) \* @PageSize + 1) and (@PageIndex \* @PageSize)

Select \* From

( Select Row\_Number() Over (Order By col1 asc ) As [\_\_IgNoRe\_\_AutoId], \* From

(

--自定义SQL，没有OrderBy

) As [\_SubQuery\_]

) As [\_\_SubQuery\_\_]

where [\_\_IgNoRe\_\_AutoId] between ((@PageIndex - 1) \* @PageSize + 1) and (@PageIndex \* @PageSize)

注： @PageIndex 索引从1 开始。

当记录数很多，查询结果在数据的后半部分时，可以反向查找：

Select \* From

( Select Row\_Number() Over (Order By col1 desc ) As [\_\_IgNoRe\_\_AutoId], \* From

(

--自定义SQL，没有OrderBy

) As [\_SubQuery\_]

) As [\_\_SubQuery\_\_]

where [\_\_IgNoRe\_\_AutoId] between (rowCount - skipNumber - takeCount + 1

) and (rowCount - skipNumber)

Order By col1 asc

#### With as 查询

with query (somcolumns ) as ( select )

select \* from query ;

with as select 必须连写使用，可以用 union。如分页，总条数是比较特殊的一行，行号可以选择 0 ， 其它列内容可为空。返回结果集后，程序需要做相应的处理。即：取第一行的 RowNumber 值当做总条数，去除第一条当做结果集。

#### 动态排序类型

示例SQL：

declare @order varchar(30)

set @order = 'asc';

select \* from Menu

order by

case @order when 'asc' then id else null end ,

case @order when 'desc' then id else null end desc

## 代码规范

类文件超过300 行，要采用分部类 。分部类文件名命名采用添加功能后缀，如：Task\_ListCard.cs

方法超过 50 行，要采用定义子方法。

页面的Hidden应该统一放在页面的上部或下部。

### 文件结构

* 报表

存储过程返回的实体类，放到 实体层 的ReportModel 文件夹。

一个报表文件分两三部分： Mvc 报表页面，WebForm 报表页面，RDLC定义文件。其中：

Mvc报表页面统一放在 Report Area 下。

WebForm 报表页面 放在 ~ /ReportWeb 文件夹下，里面的文件夹与 Mvc 报表页面Controller 统一。

RDLC 文件与 WebForm 报表页面 同级，同名。如果出现多个定义文件，则文件名添加后缀。

* 实体中列枚举

统一放到 实体层的 EntEnum 文件夹下。可以根据系统切分的子系统建子文件夹。但命名空间采用：实体名.子系统名， 也可以只采用 实体名 作为命名空间。

文件名采用 实体业务对象 + Enum

枚举名采用 实体业务对象 + 列 + Enum。其中实体业务对象是去除表前缀，后缀的名称，如：

### 命名规范

* 名称具有业务含义， 表示业务含义的名称放在前面，附加含义放到后面，如：

TaskAdd

TaskUpdate

TaskDelete

Tm\_Room.Is\_Active 列枚举是 RoomIsActiveEnum

### 代码格式

* + - 1. 对象的自动化属性，可Js的属性列表，每个属性一行，且 分隔符在行首，这样看起来比较整齐。

param = [

{ name: 'culprit', value: "flexigrid" }

, { name: 'page', value: p.newp }

, { name: 'rp', value: p.rp }

, { name: 'sortorder', value: p.sortorder || "" }

, { name: 'query', value: p.query || "" }

];

### 性能原则

避免在循环中处理资源（如数据库，文件，网络请求，Js循环中的Dom ）。

尽量减少数据库请求次数，可以按类型进行批量查询，或批量更新，或批量插入。

## 工具

### 获得弹出窗口大小

在弹出窗口打开 开发人员选项 或 FireBug，执行: jv.GetEyeSize()

### 获取SVN 的时间段历史：

http://www.cnblogs.com/rui1988/archive/2011/07/19/2113364.html#2493938

### 利用宏，自定义Copy项目中的文件：

<http://www.cnblogs.com/newsea/archive/2011/07/31/2123260.html>

### 利用宏，VS宏 之 选中解决方案中的文件：

http://www.cnblogs.com/newsea/archive/2012/09/06/2673319.html

### 利用 宏 之 转换Json数据格式。

<http://www.cnblogs.com/newsea/archive/2012/05/28/2521368.html>

## 公共页面

1. 页面生成器

页面生成器 : [~/Admin/AutoGen/Index.aspx](http://localhost/longfor/Admin/Menu/List.aspx)

页面生成器会生成基于该架构的Mvc 页面，Controller，Biz 三层代码。列表和卡片上的文字基于MyOql实体说明（xml文档），而MyOql实体说明基于数据库表和列说明，所以建议在数据库表和列上直接给出说明。

1. 配置页面

菜单管理 : [~/Admin/Menu/List.aspx](http://localhost/longfor/Admin/Menu/List.aspx)

权限配置: [~/cs/TStandardRole/List.aspx](http://localhost/longfor/cs/TStandardRole/List.aspx)

另外权限元数据配置

模块管理: [~/Admin/PowerController/List.aspx](http://localhost/longfor/Admin/PowerController/List.aspx)

页面管理: [~/Admin/PowerAction/List.aspx](http://localhost/longfor/Admin/PowerAction/List.aspx)

按钮管理: [~/Admin/PowerButton/List.aspx](http://localhost/longfor/Admin/PowerButton/List.aspx)

## WebForm 注意事项

* + - 1. aspx 页面要有 CodeBehind
      2. . csproj 项目文件 中对 webForm 文件的引用 需要以下内容，必要时手动修改：

<Compile Include="Web\db.aspx.cs">

<DependentUpon>db.aspx</DependentUpon>

<SubType>ASPXCodeBehind</SubType>

</Compile>

<Compile Include="Web\db.aspx.designer.cs">

<DependentUpon>db.aspx</DependentUpon>

</Compile>

<Content Include="Web\db.aspx" />

* + - 1. 删除反射生成的 服务器控件 变量名。使用系统生成的 .designer.cs 里的变量名。
      2. 避免 类名和 服务器控件ID 重复。 如 FORM 控件。

## WebForm 修改引用样式

1. 转移 onclick 事件到 textbox 上
2. 给 textbox 添加样式 ref。